REMARKS/ARGUMENTS

In response to the Examiner's Office Action October 5, 2007, please look at and consider the following remarks.

In regard to the rejection of claims 1 and 2 under U.S.C. 101 due to consideration of non-statutory subject matter, it should be noted that these claims have now been amended to indicate that they are not merely software per se but are part of a computer program operated by a computer.

Likewise regarding Examiner's rejection of claims 3 – 11 under 35 U.S.C. 101 on the basis the invention is directed to non-statutory subject matter, these claims have now been amended so they are not merely directed to software per se but only software which is used to operate and activate a computer system.

Examiner has rejected claims 3, 5 – 7, 9, 12, 13 as being anticipated under 35 U.S.C. 102 (e) by Srivastava (U.S. Publication no. 20030212928 A1).

At this juncture it seems that certain comments regarding the Srivastava reference A are in order here. Srivastava reference A refers to a "third party administrative agent" which is communicated with over a "GMX interface".

Here it should be indicated that Applicant's policies and parameters are "self-contained" and they are "self-actuated" on the monitored system. In the Applicant's system the users can "modify" the policies from the user interface provided in versions of the software or else they can stop the monitoring service altogether from the monitored system itself --- but there is no need or use for a "third party" administration such as is done by the Srivastava reference A.

Applicant's Health Monitor runs independently on each individual system (local system as Server 702 in Fig. 7) to be monitored. There are no additional "nodes", just other instances of the Health Monitor service also running independently on other monitored local systems.

What is monitored by Applicant's Health Monitor (Srivastava will apparently call this a subsystem) is determined dynamically by (a) whether a given application that the Health Monitor

knows how to monitor, is present in this particular system and (b) what the user has said should or should not be monitored on this particular system through the UI (User Interface).

The reference A to Srivastava mentions the use of a "administration server" where the administration server has a user interface and is configured to transmit and receive messages. But Srivastava does not discuss or explain or indicate what exactly the user interface (UI) can do. Quite contrarily, Applicant's user interface (UI) can change what is to be monitored and determine what threshold or values for each policy would constitute a health problem for a particular local system.

It should be noted and emphasized that Applicant's Health Monitor does not rule on the overall health of the system (such as Server is failed or Server is OK) . In Applicant's system each particular policy reports the status of "good/warning/failed" which is applicable only to itself and not to the entire system. Further, Applicant does not shut down the server (local system) of or any applications based on the Health Monitor status. However, Applicants may stop the monitoring of certain policies once Applicant has alerted the user that its policies seem to have failed in operation.

The reference A to Srivastava appears to be mostly focused on detecting whether to "shut down" a system or a subsystem. Quite contrarily, Applicants can detect possible problems but Applicants deliberately do not shut operations down in so as to avoid inconveniencing the users of Applicant's system unexpectedly. This is so since Applicant's code is run on what are considered to be "mission-critical" systems needing 24-hour service. Thus Applicant's philosophy is to alert the user and then let the user decide the correct course of action.

In Applicant's system, each Server such as 702 in Fig. 7 is a "local system". Each Server has its own Provider which is a data source. Typical Providers may be Event logs, Performance Counters etc. The Servers are connected to the Health Monitor Service (704) of Fig. 7 which operates to extract data from each Provider's data source. Fig. 4 indicates how 24-hour service is used to access Providers as also shown in Fig. 1 at Step 1004.

And then there is also the other problem as to whether reference A to Srivastava could provide any compatibility with a .NET operating system. It should be noted that the use by Srivastava of a "runtime MBEAN" or the use of a "JDA Subsystem" rather makes a complex and complicated system which would provide the compatibility for a .NET operating system. Thus it should be indicated that to try to combine the Srivastava system with a .NET operating system would appear to be utterly incompatible and would require a complete reengineering and redesign of the system to make such technologies compatible with each other.

The Examiner has rejected prior claims 4, 8, 10 and 14 under 35 U.S.C. 103 (a) as being obvious over Srivastava reference A in view of Wookey reference B (U.S. Patent 6,182,249 B1).

> As Examiner has indicated regarding original claim 4, it seems that Srivastava fails to state wherein said means (b) --- to create a collection includes:
>
> (b1) means to sense current operational and ability problems in each local system;
>
> (b2) means to sense future trends which can predict future problems which may occur.

Here Examiner cites Wookey reference B citing column 12 lines 14 – 26.

In looking at column 12 lines 14 – 26, it should be noted that Wookey does discuss the use of a predictive alert analysis --- but note the example that Wookey uses is --- to identify the number of memory parity errors as increasing even though the number of memory parity errors is not yet fatal.

> Likewise looking at column 12 at lines 38 Wookey discusses that the trend analysis can detect increasing disk usage and predict a problem before the threshold of 99% is reached. At line 46, it is seen Wookey uses ---- alert types define an alert in a matter similar to a token type defining a token. The alert type defines details of the alert type and how to process it.... The tokens utilizing in processing the alert included a token for the partition name.

Another example of looking at column 12, line 56 shows a predicted alert considered an alert that predicts whether or not "swap space" is going to get low in the system.

Here again it should indicated that the configuration of the Wookey technology is not compatible with the technology of the Srivastava reference A. This would require considerable redesign and reengineering in order for these two technologies to be made compatible.

It is not proper for the Examiner to pick and choose various tidy separate bits of technology from various references and then bring them together and say we have a workable system because this would not be the case.

At this point a number of further comments will be found useful regarding the Wookey reference B. The Wookey reference seems to concentrate on building up a "static tree structure" which defines all the elements of a monitored system and then will store off some diagnostic data for those elements in order to track their current status. That particular static data is then compared to some other static data which represents the desirable status of the elements so that differences are detected. As a result there is a huge amount of memory required to store data, to edit data and to pass data over modems and other elements.

Applicant's Health Monitor does not really care about or save the current data for a particular monitoring policy other than to smooth it over with several samples to avoid reacting the trend of spikes. In Applicant's system both the desired values and the current values being compared are only stored "locally" on the system being monitored --- there is no need to pass them on to another system for analysis (as is done in Wookey). Furthermore, in Applicant's system, the desired values are changeable on the fly through the user interface (UI). Likewise as items are installed or removed from the system, they are not considered to be "static" items as they are in the Wookey system.

In the Wookey reference B the tree structure is crammed full of many details for example such as (a) who is the disk manufacturer, (b) how many sectors are on a particular disk? These factors have almost nothing to do with autonomic

monitoring of disk health but is apparently intended in Wookey to let users know what sort of diagnostic tests they can run on a particular disk if they desire to do so.

In the Wookey reference B, Wookey says "the hierarchy tree can be edited in elemental hierarchy editor 215 to accommodate additions and/or delections from the hierarchy tree as when for instance, a new technology begins to be utilized in the monitored computer system. ----Quite contrarily, Applicant's Health Monitor will adjust its policy set on each individual monitored system independently if the software or hardware items that it recognizes as "monitorable" are added or removed --- here there is no need for the user to do any further manipulation.

In the Wookey reference B, it is not clear as to what triggers the comparison between the stored data for the current state --- and the data for the desired state --- does this occur only when a diagnostician requests it? Here it should be indicated that in Applicant's system, Applicant's code runs independently (does not require a request). Applicant's code raises alerts on its own as any possible problems are detected.

So again Applicants would reiterate that there is no comparability between the technology of Wookey reference B and the technology of Srivastava reference A. The recipe for a pumpkin pie is quite different from a recipe for a beef stew. These recipes are not compatible without a complete engineering redesign and reevaluation. Likewise it should be noted neither of the reference A and B can recognize any use or compatibility for a .NET operating system.

It will be noted that Applicants have amended the claims in order to provide a network which handles "local systems" (Servers) of which there are multiple number and then these local systems can be monitored and valuated for possible problems and for predictive trend analysis. This applicability is not to be found the teachings of Srivastava or in Wookey or any brochure which involves the .NET operating system.

Thus Applicants recipe for an operating network which handles local systems and which provides unusual elements of flexibility on a 24 hour basis should be seen to be a useful novel and innovative combination. In this regard it is requested that Examiner view Applicant's claims as a whole in their entirety. They should not be considered in hindsight as a cumulation of other

noted pieces of technology. Thus in this regard Applicants would now pray that Examiner note the virtue of Applicant's claims and subsequently provide a timely Notice of Allowance therefor.

Respectfully submitted,

UNISYS CORPORATION

Dated: January 4, 2008 By Alfred W. Kozak

Alfred W. Kozak
Reg. No. 24,265
Tel.: 949-380-5822
Fax: 949-380-5254

Unisys Corporation
MS 400
25725 Jeronimo Road
Mission Viejo, CA 92691

CERTIFICATE OF MAILING / TRANSMISSION (37 CFR 1.8A)

I hereby certify that this correspondence is, on the date shown below, being:

MAILING

π deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, PO Box 1450, Alexandria, VA 22313-1450.

Date January 4, 2008

FACSIMILE

☑ *transmitted by facsimile to the United States Patent and Trademark Office.*

Carol A. Wasserman January 4, 2008

Carol A. Wasserman *Date*